06-07-00

A

Please type a plus sign (+) inside this box → [+]

PTO/SB/05 (4/98)
Approved for use through 09/30/2000. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 C.F.R. § 1.53(b))

Attorney Docket No.	190-1453
First Inventor or Application Identifier	Britton
Title	Method For Generating Code For Processing A Database
Express Mail Label No.	EL 308 902 WSJ US

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231

1. [X] * Fee Transmittal Form *(e.g., PTO/SB/17)* *(Submit an original and a duplicate for fee processing)*
2. [X] Specification [Total Pages 13] *(preferred arrangement set forth below)*
   - Descriptive title of the Invention
   - Cross References to Related Applications
   - Statement Regarding Fed sponsored R & D
   - Reference to Microfiche Appendix
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawings *(if filed)*
   - Detailed Description
   - Claim(s)
   - Abstract of the Disclosure
3. [X] Drawing(s) *(35 U.S.C. 113)* [Total Sheets 2]
4. Oath or Declaration [Total Pages 3]
   a. [X] Newly executed (original or copy)
   b. [ ] Copy from a prior application (37 C.F.R. § 1.63(d)) *(for continuation/divisional with Box 16 completed)*
      i. [ ] DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. §§ 1.63(d)(2) and 1.33(b).

NOTE FOR ITEMS 1 & 13: IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28).

5. [ ] Microfiche Computer Program *(Appendix)*
6. [ ] Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
   a. [ ] Computer Readable Copy
   b. [ ] Paper Copy (identical to computer copy)
   c. [ ] Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

7. [X] Assignment Papers (cover sheet & document(s))
8. [ ] 37 C.F.R.§3.73(b) Statement [ ] Power of Attorney *(when there is an assignee)*
9. [ ] English Translation Document *(if applicable)*
10. [X] Information Disclosure Statement (IDS)/PTO-1449 [X] Copies of IDS Citations
11. [ ] Preliminary Amendment
12. [X] Return Receipt Postcard (MPEP 503) *(Should be specifically itemized)*
13. [ ] * Small Entity Statement(s) *(PTO/SB/09-12)* [ ] Statement filed in prior application, Status still proper and desired
14. [X] Certified Copy of Priority Document(s) *(if foreign priority is claimed)*
15. [ ] Other: .....

16. **If a CONTINUING APPLICATION**, *check appropriate box, and supply the requisite information below and in a preliminary amendment:*
    [ ] Continuation [ ] Divisional [ ] Continuation-in-part (CIP) of prior application No: _____/_____
    Prior application information: *Examiner* _____ *Group / Art Unit:* _____
    For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS

[ ] *Customer Number or Bar Code Label* *(Insert Customer No. or Attach bar code label here)* or [X] *Correspondence address below*

Name	William M. Lee, Jr.				
	Lee, Mann, Smith, McWilliams, Sweeney & Ohlson				
Address	P.O. Box 2786				
City	Chicago	State	Illinois	Zip Code	60690-2786
Country	USA	Telephone	(312) 368-1300	Fax	(312) 368-0034

Name (Print/Type)	William M. Lee, Jr.	Registration No. (Attorney/Agent)	26,935
Signature		Date	6/6/00

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231.

CASE NO. 190-1453
ICL

Austen John Britton et al.

SERIAL NO: To be assigned
FILING DATE: Herewith

1. Utility Patent Application Transmittal
2. Fee Transmittal for FY 1999 in duplicate
3. Assignment Transmittal and Assignment
4. Declaration and Power of Attorney
5. Claim for Priority
6. Certified Copy of UK Registration
7. Information Disclosure Statement
8. Copy of British Search Report
9. Form PTO-1449 with attached cited references
10. Check No. 45067 for $808.00
11. Specification with two (2) sheets of drawings
12. Certificate of EXPRESS MAIL.
13. Return Post Card.

DATE SENT: June 6, 2000
WJR:lmb

"Express Mail" mailing label number

EL388802651US
Date of deposit: June 6, 2000
I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Richard Sensenbrenner
(Typed or printed name of person mailing paper or fee)

Richard Sensenbrenner
(Signature of person mailing paper or fee)

METHOD FOR GENERATING CODE FOR PROCESSING A DATABASE

Background to the Invention

This invention relates to a method for generating code for processing a database, such as for example for loading a multi-dimensional data warehouse from one or more source databases.

There are already products that generate code for loading a database. In these known products, the code is generated directly from the entity-relationship model for the database. However, this approach suffers from several drawbacks. Firstly, to achieve high performance, specific tuning of the generated code is required. This makes it difficult to repeatedly build maintenance and enhancement versions of the product: typically, the directly-generated code is usable only as the starting point for a development. Secondly, the load code in practice is only part of a data warehouse solution - data extracts, automation, outside-database processing and reporting are all typical components, none of which are schema-dependent.

One object of the present invention is to provide a novel technique for generating code for processing a database, which does not suffer from these problems.

Summary of the Invention

According to the invention a method for generating code for processing a database comprises the steps:
(a) defining the database in an entity-relationship data model;
(b) creating a source file containing instructions for processing the database, the instructions including one or more high-level directives; and
(c) pre-processing the source file, by replacing the directives with code, using information pulled from the data model, to

generate a destination file containing the code for processing the database.

It can be seen that the present invention adopts a "pull" approach to the problem, instead of the conventional "push" approach. This significantly reduces the problem of tuning the code, since it gives the software developer control over the way the code is generated. Also, the "pull" approach does not restrict the developer to using a particular data modelling tool to create the data model.

Brief Description of the Drawings

Figure 1 is a schematic diagram showing a method for generating code for processing a database.

Figure 2 is a flowchart of a preprocessor used in the method.

Description of an Embodiment of the Invention

One embodiment of the invention will now be described by way of example with reference to the accompanying drawings.

It is assumed in this example that a multi-dimensional data warehouse is to be loaded from a number of individual source databases. In a retail environment, for example, the source databases may hold information about point-of-sale (POS) transactions, products, promotions, customers and stores.

Referring to Figure 1, the source databases and the data warehouse, are defined as a set of entity-relationship data models 10. These models may be produced using a commercially available modelling tool, such as ERWIN. The models include definitions of the various tables in the databases and the fields (columns) each table. The models also include additional annotations, to assist in the code generation process. These

annotations define properties of the entity (table or column), such as:

- the type of a table - {fact, summary, dimensional}
- the partitioning strategy for a table - {none, <time-key>}
- the external data type of a column.

A source file 11 defines the way in which the data warehouse is to be loaded from the individual source databases, and other operations to be performed on the data, such as cleansing and de-duplicating tables. The source file 11 is written in a combination of perl and SQL, and includes a number of special high-level commands, referred to as directives.

At system build time, a pre-processor program 12 processes the source file 11, as will be described, so as to generate a destination file 13, containing code for processing the data warehouse. The pre-processor program uses information pulled from the data models 10. It also uses a set of include files 14 and template files 15, as will be described. The include files and the template files may be written in a combination of perl and SQL, or possibly any other programming language, and may also include directives.

Directives

The name of each directive is prefixed by the symbol # to identify it as such to the pre-processor. By way of example, the directives may include the following.

#aggregate_table This directive instructs the pre-processor to generate SQL code to aggregate data from a specified source table into a specified destination table.

#append_table This directive instructs the pre-processor to generate SQL code to append rows from a specified source table to a specified destination table.

#cleanse_table This directive instructs the pre-processor to generate SQL code to perform cleansing of a specified source table, including supply of default values, deletion of irrelevant columns and addition of new columns. The result of the processing is placed in a specified destination table.

#create_table This directive instructs the pre-processor to generate SQL code to create a table with a specified name. Optionally, it may specify that the table should have the same database schema as an existing table.

#deduplicate_table This directive instructs the pre-processor to generate SQL code to deduplicate a specified source table (i.e. to remove records that share the same primary key, so that there remains only one record with any given primary key). The result of the processing is placed in a specified destination table.

#define This directive defines a macro, i.e. a text substitution which is used throughout the rest of the pre-processing. For example, the directive "#define INTSIZE 4" indicates that the value "4" is to be substituted for "INTSIZE". Macros may contain directives and nested macro calls. A macro, once declared, never goes out of scope.

#include This directive instructs the pre-processor to include code from a specified one of the include files 14.

#prefix This directive instructs the pre-processor to modify any word beginning with a specified prefix, by replacing the prefix with a specified new prefix.

Prefixes allow SQL variants in naming standards, such as standards for naming temporary variables, to be accommodated. For example, a prefix "TT_" may be defined as "#" for a first database, and as "" (i.e. nothing) for a second database. All occurrences of "TT_<table_name>" will then be converted by the pre-processor to either "#<table_name>" or "<table_name>", depending on the target database.

#template This directive instructs the pre-processor to include code from a specified one of the template files 15. Templates allow SQL variants that use source constructs in different orders to be accommodated. The #template directive supplies the sets of text to be inserted, and the template file itself supplies the keywords and the variant-specific ordering.

For example, the ordering of the clauses within an SQL INSERT statement to insert records into a table varies between databases. The #template directive allows the developer to specify an INSERT statement independently of the target database, by means of the following code:

```
#template<template name>
#INSERT_INTO
<insert-text set>
#SELECT
<select-text set>
#FROM
<from-text set>
#end_template
```

Pre-processor

The pre-processor program 12 will now be described in more detail, with reference to Figure 2.

The pre-processor program steps through the source file 11, a line at a time. For each line, the pre-processor checks whether

the line contains a directive. If a directive is found, it is processed, as follows:

- In the case of a directive that generates SQL code (such as #aggregate_table) the pre-processor accesses the data models 10 to pull information from them about the structure of the source and destination tables, and to read any relevant annotations. It then uses this information to generate the required SQL code. The pre-processor also inserts comments in the generated code, to make it easier for the software developer to understand and modify the code. The generated code is written to a temporary file, and the preprocessor then recursively calls itself, to process this temporary file.

- In the case of a #define directive, the macro definition is added to a list of macro definitions.

- In the case of a #prefix directive, the prefix is added to a list of prefixes to be substituted.

- In the case of an #include or #template directive, the pre-processor temporarily stops processing the file currently being processed, and recursively calls itself to process the specified include file or template file.

If, on the other hand, the line does not contain a directive, the pre-processor proceeds as follows.

First, the pre-processor looks for any prefixes in the line, and tests them against the list of prefixes. If a prefix is listed, the required prefix substitution is performed.

Next, the pre-processor splits the line into words, by identifying a predetermined group of characters on which words should be broken. This group includes spaces, tabs and newlines

as well as quotes, brackets and a number of other non-word characters.

Each word is then tested against the list of macro definitions that have been declared. If a macro has been declared for any of the words, the macro is expanded, by substituting the specified replacement text.

If any macro substitutions have been made, the expanded line is written to a temporary file. The pre-processor then recursively calls itself, to process the temporary file. Thus, nested macros (including directives within macros) can be expanded. When the temporary file has been processed, it is deleted.

If, on the other hand, no macro substitutions were made, the line is simply appended to the destination file 13.

The output of the pre-processor 11 is a destination file 13, which contains the SQL code for loading and performing other processing operations on a database. This file is complete and ready to run. If desired, critical sections of the code may be hand-crafted, rather than by using directives to generate them, so as to achieve higher performance.

Run-time processing

As stated, the output of the pre-processor is a file containing SQL code. However, some of the requirements for the SQL code may not be known until run-time (i.e. until the code is actually executed). For example, the way in which the data is to be physically stored on disks may not be known, since the database storage strategy may regularly be changed and refined. This problem is handled as follows.

Whenever a directive, such as #create_table, generates SQL code to explicitly create a table, the pre-processor inserts a

special macro, referred to herein as an run-time processor (rtp) macro, in the place of the storage clause. The storage clause describes how the data in the table is to be physically stored. An rtp macro can specify a list of storage scheme identities (some of which may be non-existent). The storage scheme identities are listed in order of preference.

A storage scheme file 16 is provided, containing a set of current storage schemes. This file can be modified by the end-user as required. For example, if the user adds more disks to the system or changes the database storage strategy, the storage schemes may be updated to reflect this.

At run-time, a run-time processor 17 scans the code, looking for rtp macros. When it finds an rtp macro, it searches the storage scheme file 16, to find a storage scheme specified in the macro, starting with the most preferred scheme. If none of the storage schemes specified in the macro exists, a default scheme is chosen. The selected scheme is then used to generate a storage clause, which is added to the code.

The use of rtp macros in this way means that specification of the physical storage of the data can be deferred until run-time; in other words, it is performed "just in time". This avoids the need for having to re-build the entire code every time the storage strategy changes: only the storage scheme file needs to be changed.

Some possible modifications

It will be appreciated that many modifications may be made to the system described above without departing from the scope of the present invention. For example, other languages such as C or Java may be used in the source files, include files and template files. Also, other modelling tools may be used to generate the data models.

Although the run-time processor has been described in relation to storage schemes, its technology could also be applied to any other task that needs to be completed just in time.

Conclusion

In summary, it can be seen that the method described above provides a way of generating code for loading data into multi-dimensional databases which enables high performance, flexible solutions to be developed for different database models, using different relational databases running on different operating systems, from a single common technology. It addresses the requirement for a method of rapid, reliable generation of high-performance code to load data into such databases.

Experience has shown that using directives as described can reduce the size of the code required by a factor of 15, averaged over a whole project. In some areas the factor is around 35. Additional benefits are extremely cost-effective development and ongoing support.

CLAIMS

1. A method for generating code for processing a database comprising the steps:
(a) defining the database in an entity-relationship data model;
(b) creating a source file containing instructions for processing the database, the instructions including one or more high-level directives; and
(c) pre-processing the source file, by replacing the directives with code, using information pulled from the data model, to generate a destination file containing the code for processing the database.

2. A method according to Claim 1 wherein at least some of the directives define macro substitutions to be performed on the source file.

3. A method according to Claim 1 wherein at least some of the directives specify another file containing code to be included in the destination file.

4. A method according to Claim 1 wherein at least some of the directives cause run-time macros to be inserted into the destination file, and wherein the method includes the step of processing the destination file at run time to replace the run-time macros.

5. A method according to Claim 4 wherein the run-time macros define a storage scheme for the database.

6. A method according to Claim 1 wherein the model includes annotations, and wherein the method includes using the annotations to control pre-processing of the source file.

7. A method for processing a database comprising the steps:
(a) defining the database in an entity-relationship data model;
(b) creating a source file containing instructions for processing the database, the instructions including one or more high-level directives;
(c) pre-processing the source file, by replacing the directives with code, using information pulled from the data model, to generate a destination file containing the code for processing the database; and
(d) running the code in the destination file, to process the database.

8. A method according to Claim 7 wherein at least some of the directives define macro substitutions to be performed on the source file.

9. A method according to Claim 7 wherein at least some of the directives specify another file containing code to be included in the destination file.

10. A method according to Claim 7 wherein at least some of the directives cause run-time macros to be inserted into the destination file, and wherein the method includes the step of processing the destination file at run time to replace the run-time macros.

11. A method according to Claim 10 wherein the run-time macros define a storage scheme for the database.

12. A method according to Claim 7 wherein the model includes annotations, and wherein the method includes using the annotations to control pre-processing of the source file.

13. A computer system comprising:
(a) means for defining a database in an entity-relationship data model;
(b) means for creating a source file containing instructions for processing the database, the instructions including one or more high-level directives;
(c) means for pre-processing the source file, by replacing the directives with code, using information pulled from the data model, to generate a destination file containing the code for processing the database; and
(d) means for running the code in the destination file, to process the database.

14. A computer system according to Claim 13, wherein at least some of the directives cause run-time macros to be inserted into the destination file, and wherein the system includes means for processing the destination file at run time to replace the run-time macros.

15. A computer system according to Claim 13 wherein the model includes annotations, and wherein the system includes means for using the annotations to control pre-processing of the source file.

16. An information carrier, holding a program for performing a method for generating code for processing a database defined in an entity-relationship data model, the method comprising the steps:
(a) creating a source file containing instructions for processing the database, the instructions including one or more high-level directives; and
(b) pre-processing the source file, by replacing the directives with code, using information pulled from the data model, to generate a destination file containing the code for processing the database.

ABSTRACT

A method is described for generating code for loading and performing other processing operations on a multi-dimensional data warehouse. Databases are defined as entity-relationship data models, using a modelling tool. A source file is created, containing instructions for processing the database, and including one or more high-level directives. At system build time, the source file is pre-processed, by replacing the directives with code, using information pulled from the data model, to generate a destination file containing the code for processing the database. At run time, the generated code is processed to replace run-time macros, indicating physical storage schemes. The generated code is then run, to process the database.

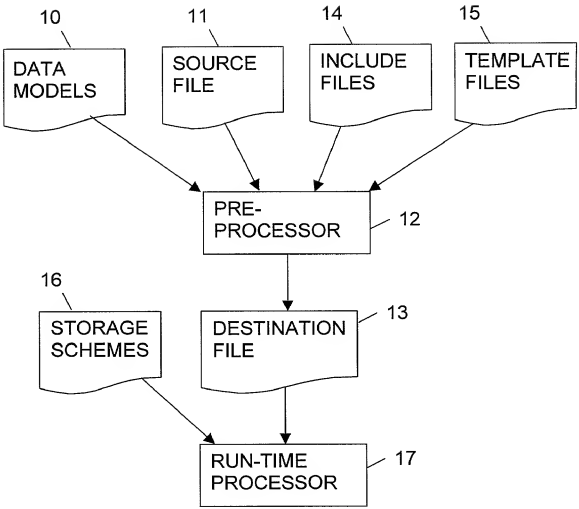
10
DATA MODELS
11
SOURCE FILE
14
INCLUDE FILES
15
TEMPLATE FILES
PRE-PROCESSOR
12
16
STORAGE SCHEMES
DESTINATION FILE
13
RUN-TIME PROCESSOR
17

FIG.1

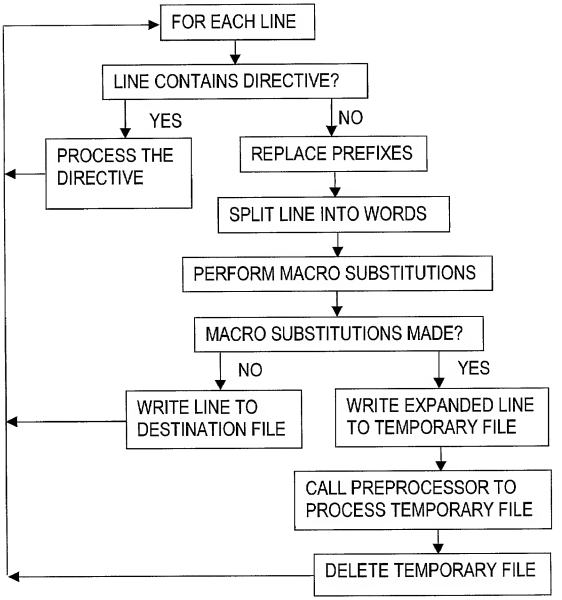
FOR EACH LINE
LINE CONTAINS DIRECTIVE?
YES
NO
PROCESS THE DIRECTIVE
REPLACE PREFIXES
SPLIT LINE INTO WORDS
PERFORM MACRO SUBSTITUTIONS
MACRO SUBSTITUTIONS MADE?
NO
YES
WRITE LINE TO DESTINATION FILE
WRITE EXPANDED LINE TO TEMPORARY FILE
CALL PREPROCESSOR TO PROCESS TEMPORARY FILE
DELETE TEMPORARY FILE

FIG.2

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

Method for Generating Code for Processing a Database

the specification of which:

X is attached hereto.

___ was filed on _____

as

Application Serial No.

and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

PRIOR FOREIGN APPLICATION(S)

			Priority Claimed	
Country	**Number**	**Date Filed**	**Yes**	**No**
United Kingdom	GB 9914405.7	22 Jun 1999	X	___
_____	_____	_____	___	___
_____	_____	_____	___	___

I hereby claim the benefit under Title 35, United States Code Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application.

Application Serial No.	**Filing Date**	**Status**

And I hereby appoint Wm. Marshall Lee, Registration No. 16,853, John M. Mann, Registration No. 17,775, Thomas E. Smith, Registration No. 18,243, Dennis M. McWilliams, Registration No. 25,195, James R. Sweeney, Registration No. 18,721, William M. Lee, Jr., Registration No. 26,935, Glenn W. Ohlson, Registration No. 28,455, David C. Brezina, Registration No. 34,128, Jeffrey R. Gray, Registration No. 33,391, Timothy J. Engling, Registration No. 39,970, Gregory B. Beggs, Registration No. 19,286, Gerald S. Geren, Registration No. 24,528 and Peter J. Shakula, Registration No. 40,808 as my attorneys to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith. It is requested that all communications be directed to Lee, Mann, Smith, McWilliams, Sweeney & Ohlson, P.O. Box 2786, Chicago, Illinois 60690-2786, telephone number (312) 368-1300.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: **Austen John Britton**

Signature _____ Date 20 April 2000
Country of Residence: United Kingdom
Country of Citizenship: United Kingdom
Post Office and Residence Address: 28 Turncliff Crescent, Marple, Stockport, Cheshire, SK6 6JP, England

Full name of joint inventor: **Margaret Hazel Derbyshire**

Signature Margaret M Derbyshire Date 8 May 2000
Country of Residence: United Kingdom
Country of Citizenship: United Kingdom
Post Office and Residence Address: 2a King's Mead, Ripon, North Yorkshire, HG4 1EJ, England

Full name of joint inventor: **Mark Hinton**

Signature _____ Date 27th April 2000
Country of Residence: United Kingdom
Country of Citizenship: United Kingdom
Post Office and Residence Address: 3 Lyme Grove, Marple, Cheshire, SK6 7NW, England

Full name of joint inventor: **David Stephen Walsh**

Signature David Stephen Walsh Date 5th May 2000
Country of Residence: United Kingdom
Country of Citizenship: United Kingdom
Post Office and Residence Address: 24 Billington Avenue, Rawtenstall, Lancashire, England